Assignment 1:

MVC is a software architectural pattern that separates an application into three interconnected components: Model, View, and Controller. Each architectural component is built to handle specific development aspects of an application. This separation enables the independent development, testing, and maintenance of each component, improving code readability, reusability, and scalability.

Model:

- Represents the application's data and business logic.
- Responsible for managing data storage, retrieval, and manipulation.
- Does not directly interact with the user interface.

View:

- Presents the data to the user.
- Renders the user interface elements based on the data received from the Model.
- Typically passive and doesn't directly handle user input.

Controller:

- Acts as an intermediary between the Model and the View.
- Receives user input and invokes appropriate actions on the Model.
- Updates the View with the changes in the Model.

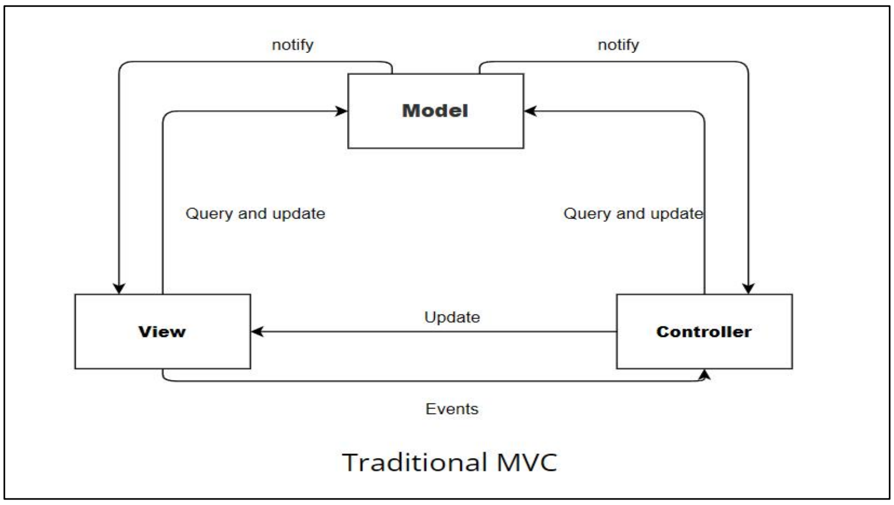
Variants:

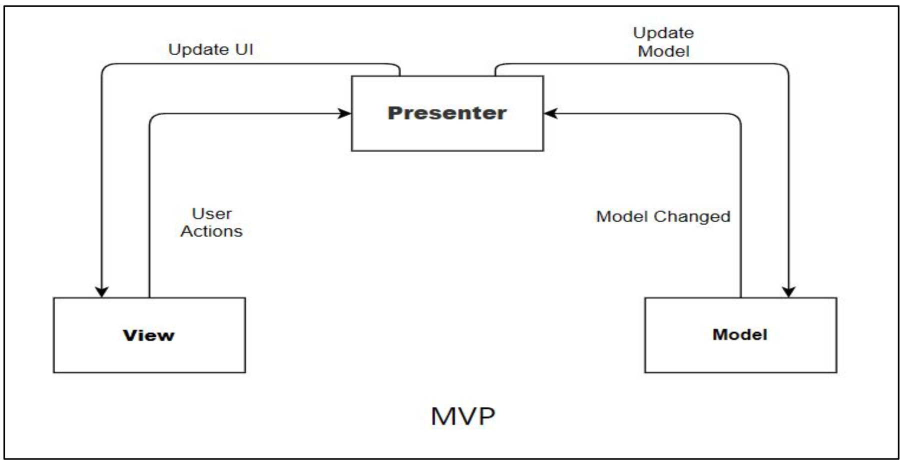
Traditional MVC:

- Model directly notifies the View of any changes.
- Controller handles user input and updates the Model accordingly.
- Suitable for desktop applications and smaller-scale projects.

Model-View-Presenter (MVP):

- Presenter acts as an intermediary between the View and the Model.
- View delegates user input handling to the Presenter.
- Presenter updates the View directly and modifies the Model.
- Suitable for large-scale projects with complex user interfaces


notify
notify
Model
Query and update
Query and update
View
Update
Controller
Events
Traditional MVC


Update UI
Update Model
Presenter
User Actions
Model Changed
View
Model
MVP

Assignment2:

Scenario: Applying Microservices and Event-Driven Architecture to E-commerce

Approach:

- Break down e-commerce into Microservices: Product Catalog, Order Management, etc.
- Utilize Event-Driven Architecture for asynchronous communication.
- Adhere to SOLID principles for maintainability.

SOLID Principles:

- SRP: Each service has one responsibility.
- OCP: Open for extension, closed for modification.
- LSP: Services can be swapped if they adhere to the same interface.
- ISP: Design specific interfaces for clients.
- DIP: High-level modules depend on abstractions.

Observing DRY and KISS:

- DRY: Centralize common functionalities, avoid code duplication.
- KISS: Keep services simple, solving one problem efficiently.

Outcome:

Efficient, scalable, and maintainable e-commerce platform meeting modern architectural standards.

Assignment 4:
Trends and Cloud Services

Serverless Architecture Benefits:

Serverless architecture offers numerous benefits, including reduced operational complexity, scalability, and cost-effectiveness. With serverless computing, developers can focus solely on writing code without worrying about server management. The pay-per-use model ensures cost efficiency by charging only for the resources consumed. Additionally, serverless architecture automatically scales up or down based on demand, ensuring optimal performance and eliminating the need for capacity planning.

Progressive Web Apps (PWAs):

Progressive Web Apps (PWAs) combine the best features of web and mobile applications, offering a seamless user experience across devices. PWAs utilize modern web technologies to provide fast loading times, offline capabilities, and push notifications. By leveraging service workers and caching, PWAs can deliver app-like experiences without the need for installation from an app store. This approach enhances user engagement and retention while reducing development and maintenance overhead.

AI and Machine Learning in Software Architecture:

AI and machine learning are revolutionizing software architecture by enabling intelligent decision-making, automation, and personalization. AI-powered algorithms can analyze vast amounts of data to extract insights, detect patterns, and make predictions, enhancing application functionality. Machine learning models can optimize resource allocation, improve security through anomaly detection, and enhance user experience through personalized recommendations. Integrating AI and machine learning into software architecture empowers applications to adapt and evolve in response to changing requirements and user behaviors.

Cloud Computing Service Models Overview:

Cloud computing offers three primary service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). SaaS delivers software applications over the internet on a subscription basis, eliminating the need for installation and maintenance. PaaS provides a platform for developers to build, deploy, and manage applications without worrying about infrastructure. IaaS offers virtualized computing resources such as servers, storage, and networking on a pay-as-you-go basis, allowing users to deploy and manage their applications and services. Each service model has its unique use cases, catering to different requirements and preferences of businesses and developers.